

Crime Analysis Database

- Class : DAMG6210
- CRN : 16149
- Team Name: Crime Board
- Team members:
 - Sayeed Ahmed - 002191535
 - Natarajan Lekshmi Narayana Pillai - 002766033
 - Chaman Betrabet - 002784662
 - Sunil Rudrakumar - 002764807

Summary

The objective of our database is to have a centralized data source that would cater to any crime analysis system. It would be able to provide an efficient database for various use cases such as:

- Real Estate - Provide a database to have data visualization for the crime rate in a locality.
- Law Enforcement - Provide law enforcement a source to gather and track the various crimes around their jurisdiction.
- Location Service - Navigation services such as Google Maps can have an option to provide the safest route from Point A to Point B.
- Tourist Destinations - Make tourists aware of a particular area's safety before they visit.

The data would be collected through various sources to collate and create a new database that would be able to answer questions on the data like how crime has progressed, what type of crimes take place, and their severity and provide a correlation between various factors(gross income of an area, time, etc) and crime committed, etc.

Steps taken:

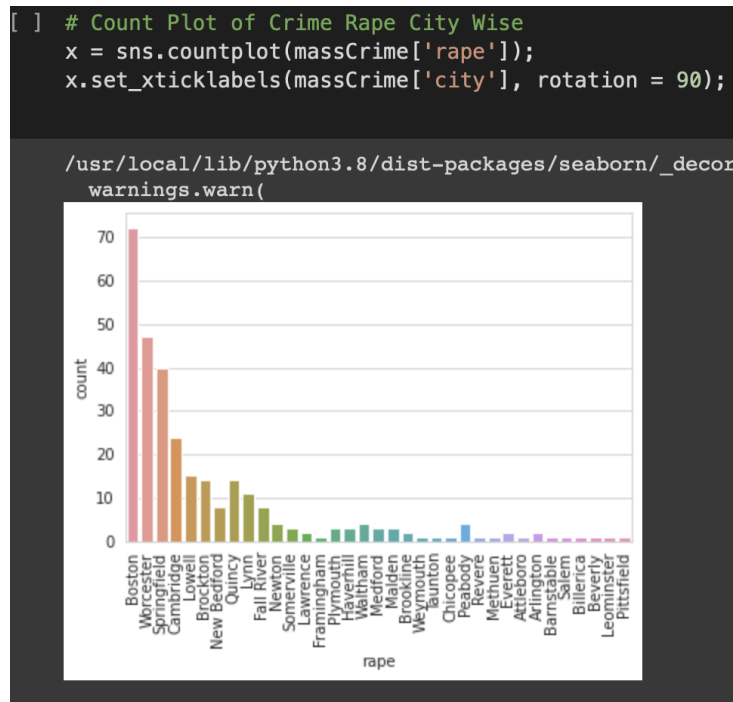
Sourcing Data:

- 1) Twitter: We collated tweets data to gather various incidents or conversations regarding crime. We did this by using a twitter scrapper bot coded in python that would crawl for specific key words regarding to crime like murder, rape, gunshot etc and would gather the various details like tweet text, meta data of the tweet like tweet id, like count etc and also the user details.
- 2) Crime data: For the crime data we used the official datasets from the Boston Police department.

Munging and cleaning the data:

- We imported the data sets as csv files and parse it using the pandas library in python.
- Once the data was made into a data frame, we utilized the various transformation functions available in the pandas library to handle the data issues like incomplete, duplicates and corrupted data.

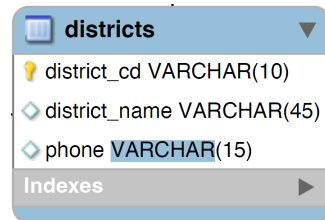
Auditing the data:



After cleaning the data, we analyzed it and discovered some useful insights, such as the fact that Boston has the highest number of crimes in all categories in the state of Massachusetts.

Loading the data into a database:

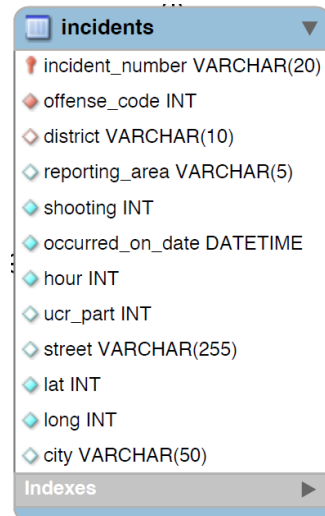
- Once the data was cleaned and validated, we loaded the transformed data frames into a MySQL database system.
- The data was loaded into the following tables:
 - Districts : The table contained the various districts in and around boston.



The screenshot shows the 'districts' table with the following columns: district_cd VARCHAR(10), district_name VARCHAR(45), and phone VARCHAR(15). There is an 'Indexes' section at the bottom with a right-pointing arrow.

Column	DataType
district_cd	VARCHAR(10)
district_name	VARCHAR(45)
phone	VARCHAR(15)

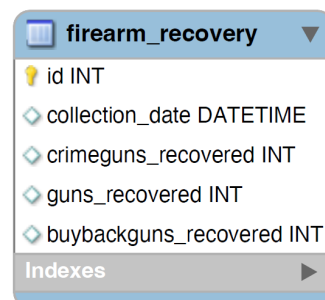
- Incidents: The table contains the incident reports by the BDP.



The screenshot shows the 'incidents' table with the following columns: incident_number VARCHAR(20), offense_code INT, district VARCHAR(10), reporting_area VARCHAR(5), shooting INT, occurred_on_date DATETIME, hour INT, ucr_part INT, street VARCHAR(255), lat INT, long INT, and city VARCHAR(50). There is an 'Indexes' section at the bottom with a right-pointing arrow.

Column	DataType
incident_number	VARCHAR(20)
offense_code	INT
district	VARCHAR(10)
reporting_area	VARCHAR(5)
shooting	INT
occurred_on_date	DATETIME
hour	INT
ucr_part	INT
street	VARCHAR(255)
lat	INT
long	INT
city	VARCHAR(50)

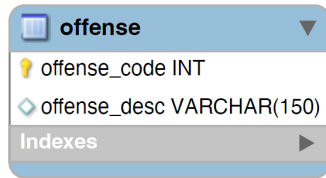
- Firearm_recovery: The table contains the various firearms recovered during a crime bust and also that were turned in or retired.



The screenshot shows the 'firearm_recovery' table with the following columns: id INT, collection_date DATETIME, crimeguns_recovered INT, guns_recovered INT, and buybackguns_recovered INT. There is an 'Indexes' section at the bottom with a right-pointing arrow.

Column	DataType
id	INT
collection_date	DATETIME
crimeguns_recovered	INT
guns_recovered	INT
buybackguns_recovered	INT

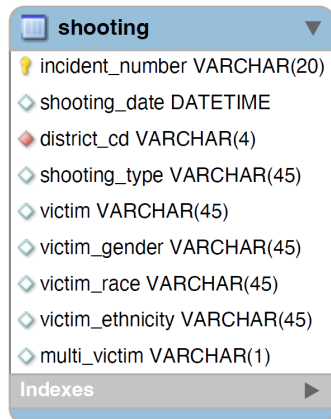
- Offense: It contains the data regarding the official offense codes and their descriptions



The screenshot shows the 'offense' table with two columns: 'offense_code' of type INT and 'offense_desc' of type VARCHAR(150). There is an 'Indexes' section at the bottom with a right-pointing arrow.

offense	
offense_code	INT
offense_desc	VARCHAR(150)
Indexes ▶	

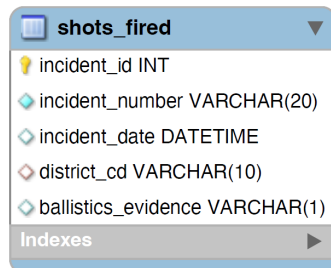
- Shooting: it contains data regarding the incidents that involved shootings and various parameters related to it.



The screenshot shows the 'shooting' table with nine columns: 'incident_number' (VARCHAR(20)), 'shooting_date' (DATETIME), 'district_cd' (VARCHAR(4)), 'shooting_type' (VARCHAR(45)), 'victim' (VARCHAR(45)), 'victim_gender' (VARCHAR(45)), 'victim_race' (VARCHAR(45)), 'victim_ethnicity' (VARCHAR(45)), and 'multi_victim' (VARCHAR(1)). There is an 'Indexes' section at the bottom with a right-pointing arrow.

shooting	
incident_number	VARCHAR(20)
shooting_date	DATETIME
district_cd	VARCHAR(4)
shooting_type	VARCHAR(45)
victim	VARCHAR(45)
victim_gender	VARCHAR(45)
victim_race	VARCHAR(45)
victim_ethnicity	VARCHAR(45)
multi_victim	VARCHAR(1)
Indexes ▶	

- Shots_fired: it contains the data related to reports of gunshots that have been recorded from various sources.



The screenshot shows the 'shots_fired' table with six columns: 'incident_id' (INT), 'incident_number' (VARCHAR(20)), 'incident_date' (DATETIME), 'district_cd' (VARCHAR(10)), and 'ballistics_evidence' (VARCHAR(1)). There is an 'Indexes' section at the bottom with a right-pointing arrow.

shots_fired	
incident_id	INT
incident_number	VARCHAR(20)
incident_date	DATETIME
district_cd	VARCHAR(10)
ballistics_evidence	VARCHAR(1)
Indexes ▶	

- Tweets: it contains the data related to tweets that were scrapped using the twitter bot

tweets	
id_str	DECIMAL(20,0)
text	VARCHAR(1000)
created_at	DATETIME
retweet_count	INT
favorite_count	INT
city	VARCHAR(100)
state	VARCHAR(100)
country	VARCHAR(100)
id	DECIMAL(20,0)
Indexes	

- Tweet_userhist: it is used to record the user activity of the users in the tweets table and record the tweets within the last 24 hours.

tweet_userhist	
user_id	DECIMAL(20,0)
tweet_text	VARCHAR(1000)
Indexes	

- Tweet_users: it is used to record the user's data like user id, followers etc.

tweet_users	
id	DECIMAL(20,0)
screen_name	VARCHAR(100)
followers_count	INT
created_at	DATETIME
Indexes	

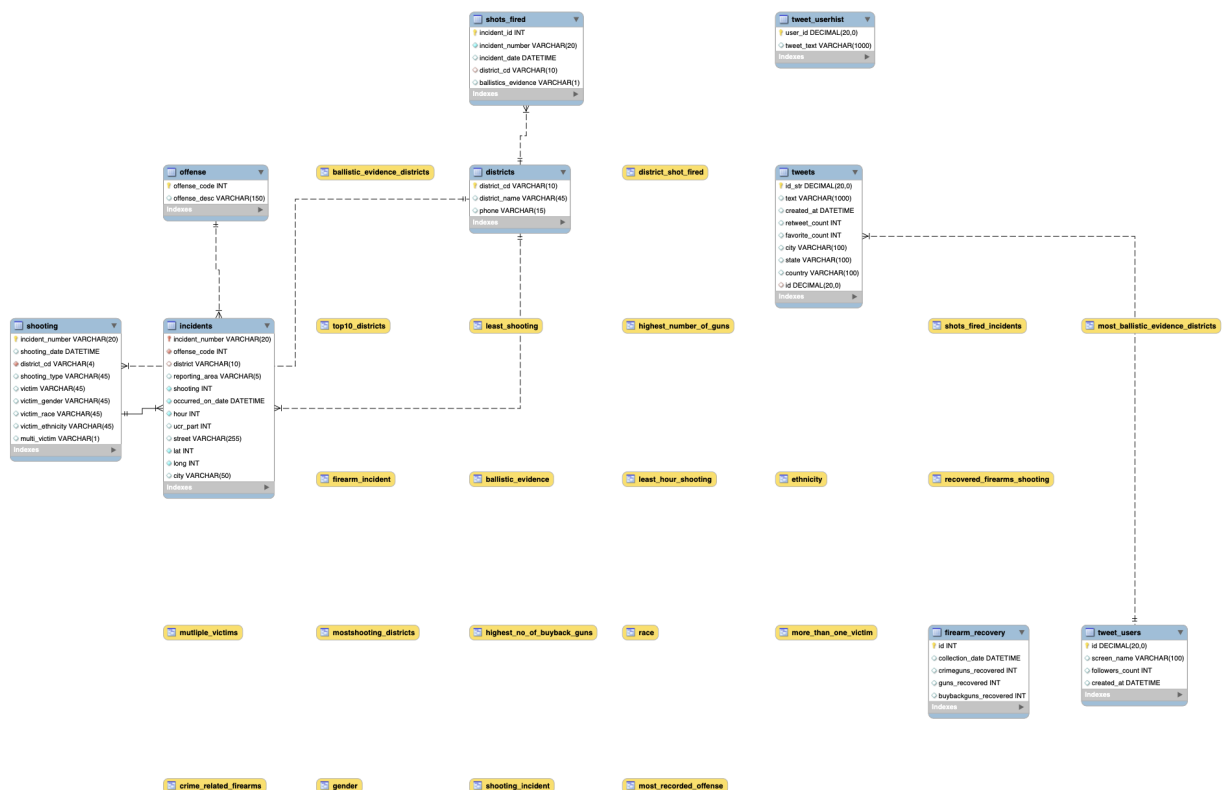
Normalization:

We have done the normalization for the tables. For converting to 1st Normal Form (1NF), we have identified a unique primary key for all the tables. Additionally we removed columns having atomic values and eliminated repeating groups.

For 2nd Normal Form (2NF), we have ensured the database is in first normal form. Then we removed all partial dependencies between the primary keys. For this we ensured that the offense description field, which was initially in the incidents table, was eliminated as it could be linked to the offense master table, which has the primary key as `offense_code`. Additionally we removed attributes like day incident occurred on, as it can be derived from the date field already available in the incidents table.

Our Database was mostly in 3rd Normal Form (3NF), and we ensured that there were no transitive dependencies between any fields in our tables. Please find below the ER Diagram after normalization was done:

ER Diagram After Normalization.

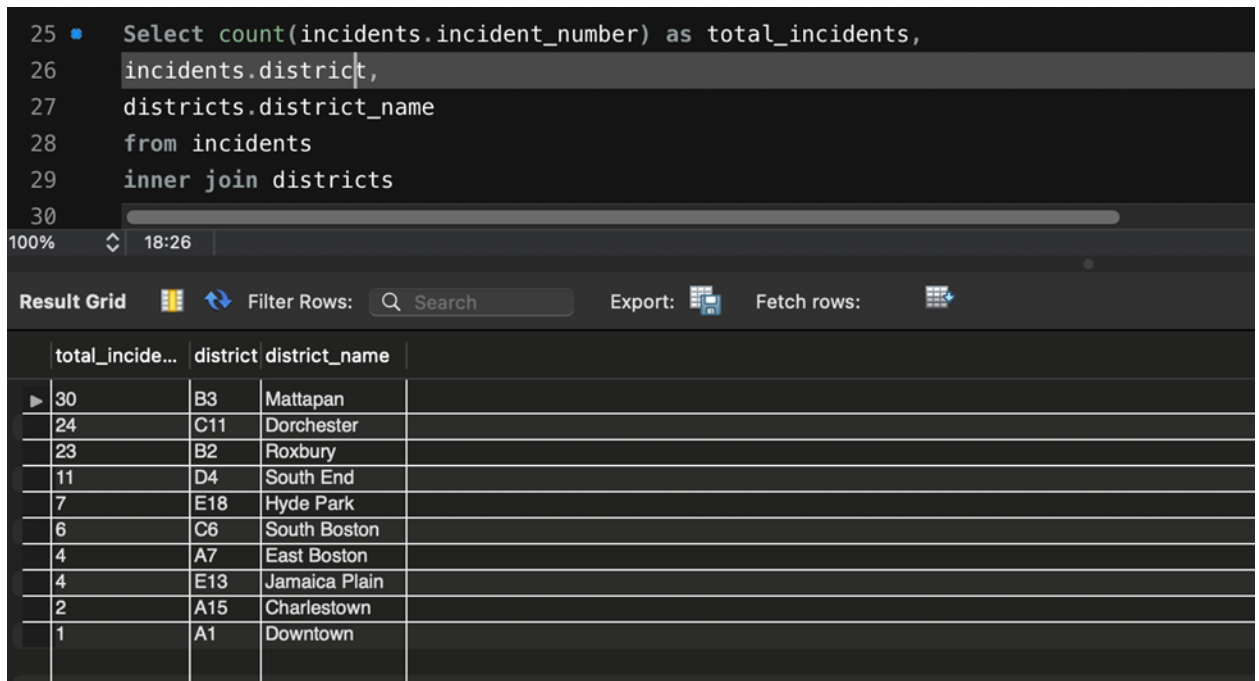


USE CASES

Please find below some use cases which we have displayed:

1. What are the top 10 districts that have the most incident reports?

```
Select count(incidents.incident_number) as total_incidents,  
incidents.district,  
districts.district_name  
from incidents  
inner join districts  
on incidents.district = districts.district_cd  
group by incidents.district  
order by total_incidents desc  
limit 10;
```



The screenshot shows a SQL query editor with a dark theme. The query is as follows:

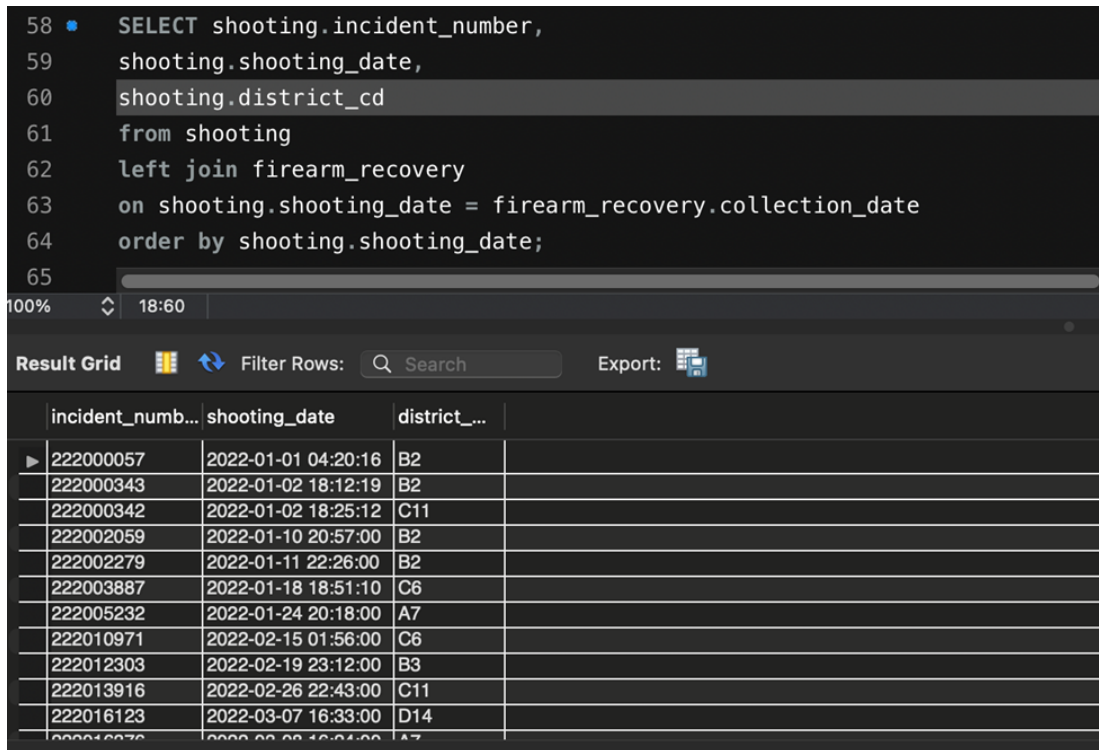
```
25 • Select count(incidents.incident_number) as total_incidents,  
26 incidents.district,  
27 districts.district_name  
28 from incidents  
29 inner join districts  
30
```

Below the query editor is a 'Result Grid' showing the top 10 districts by incident count. The grid has columns for 'total_incidents', 'district', and 'district_name'. The results are sorted in descending order of incident count.

	total_incide...	district	district_name
▶	30	B3	Mattapan
	24	C11	Dorchester
	23	B2	Roxbury
	11	D4	South End
	7	E18	Hyde Park
	6	C6	South Boston
	4	A7	East Boston
	4	E13	Jamaica Plain
	2	A15	Charlestown
	1	A1	Downtown

2. Were firearms recovered on the dates when shootings occurred?

```
SELECT shooting.incident_number,  
       shooting.shooting_date,  
       shooting.district_cd  
from shooting  
left join firearm_recovery  
on shooting.shooting_date = firearm_recovery.collection_date  
order by shooting.shooting_date;
```



The screenshot shows a SQL query editor with a dark theme. The query is as follows:

```
58 SELECT shooting.incident_number,  
59 shooting.shooting_date,  
60 shooting.district_cd  
61 from shooting  
62 left join firearm_recovery  
63 on shooting.shooting_date = firearm_recovery.collection_date  
64 order by shooting.shooting_date;  
65
```

Below the editor is a 'Result Grid' showing the results of the query. The grid has four columns: 'incident_num...', 'shooting_date', 'district_...', and an empty column. The results are sorted by 'shooting_date'.

	incident_num...	shooting_date	district_...	
▶	222000057	2022-01-01 04:20:16	B2	
	222000343	2022-01-02 18:12:19	B2	
	222000342	2022-01-02 18:25:12	C11	
	222002059	2022-01-10 20:57:00	B2	
	222002279	2022-01-11 22:26:00	B2	
	222003887	2022-01-18 18:51:10	C6	
	222005232	2022-01-24 20:18:00	A7	
	222010971	2022-02-15 01:56:00	C6	
	222012303	2022-02-19 23:12:00	B3	
	222013916	2022-02-26 22:43:00	C11	
	222016123	2022-03-07 16:33:00	D14	
	222016270	2022-03-08 16:04:00	A7	

3. Which gender was involved with the most shooting?

```
Select shooting.victim_gender,  
count(shooting.victim_gender) as count,  
incidents.offense_description  
from shooting  
inner join incidents  
on shooting.incident_number = incidents.incident_number  
group by shooting.victim_gender, incidents.offense_description;
```

```
75 • Select shooting.victim_gender,  
76     count(shooting.victim_gender) as count,  
77     incidents.offense_description  
78     from shooting  
79     inner join incidents  
80     on shooting.incident_number = incidents.incident_number  
81     group by shooting.victim_gender, incidents.offense_description;
```

100% 23:76

Result Grid Filter Rows: Search Export:

	victim_gend...	count	offense_description
▶	Male	70	ASSAULT - AGGRAVATED
	Female	2	MURDER, NON-NEGLIGIENT MANSLAUGHTER
	Female	16	ASSAULT - AGGRAVATED
	Male	2	MURDER, NON-NEGLIGIENT MANSLAUGHTER
	Male	3	ROBBERY
	Male	20	MURDER, NON-NEGLIGENT MANSLAUGHTER
	Male	1	DEATH INVESTIGATION
	Female	1	MURDER, NON-NEGLIGENT MANSLAUGHTER

4. Which type of offense is the most recorded?

```
Select incidents.offense_code,
count(incidents.offense_code) as count_of_offence,
offense.offense_desc
from incidents
Join offense
on incidents.offense_code = offense.offense_code
group by offense_code
order by count_of_offence;
```

```
99  *   Select incidents.offense_code,
100      count(incidents.offense_code) as count_of_offence,
101      offense.offense_desc
102  from incidents
103  Join offense
104  on incidents.offense_code = offense.offense_code
105  group by offense_code
106  order by count_of_offence;
107
```

100% 13:103

Result Grid



Filter Rows:

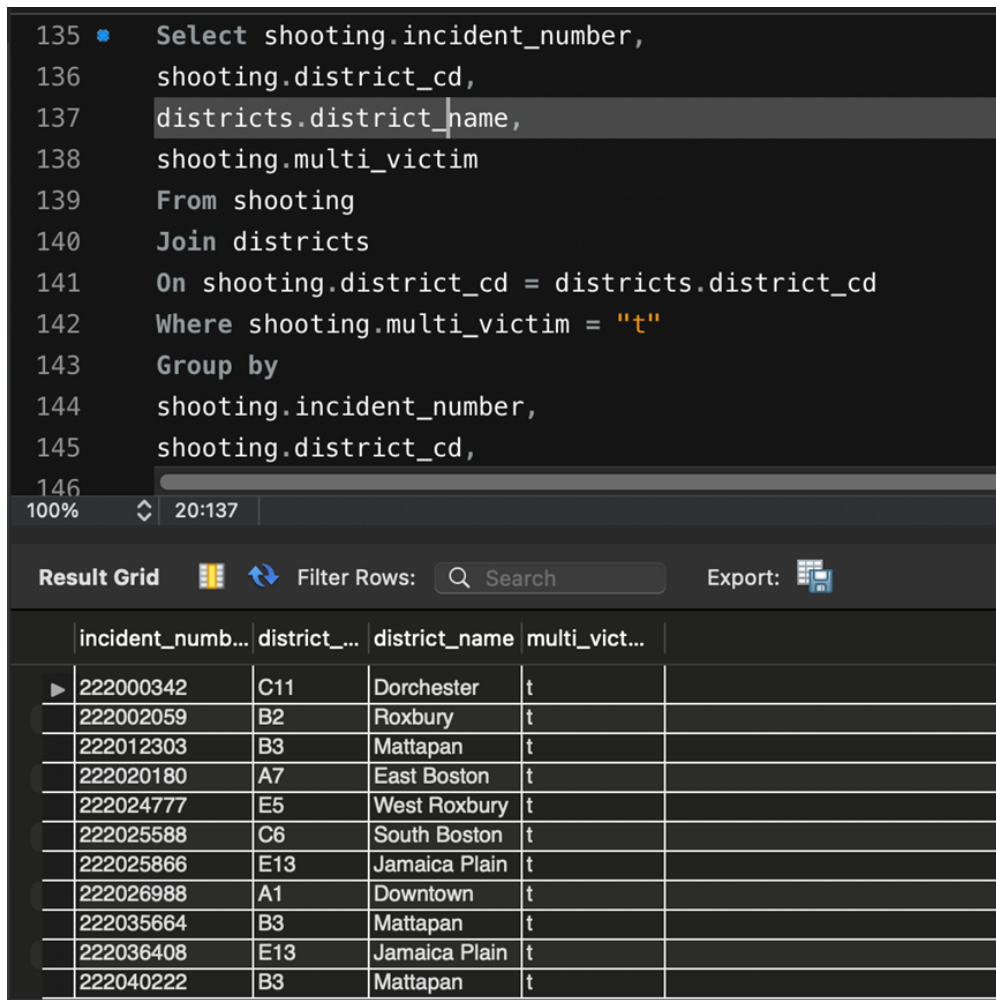


Export:

[illegible]

5. Which district had more than 1 victim?

```
Select shooting.incident_number,  
       shooting.district_cd,  
       districts.district_name,  
       shooting.multi_victim  
From shooting  
Join districts  
On shooting.district_cd = districts.district_cd  
Where shooting.multi_victim = "t"  
Group by  
       shooting.incident_number,  
       shooting.district_cd,  
       districts.district_name,  
       Shooting.multi_victim;
```



The screenshot shows a SQL query editor with a query that selects incident numbers, district codes, district names, and multi-victim status from a shooting table, joined with a districts table. The query is filtered for multi-victim incidents (multi_victim = 't') and grouped by incident number, district code, district name, and multi-victim status. Below the query editor, the 'Result Grid' is displayed, showing 14 rows of data. The columns are incident_number, district_cd, district_name, and multi_victim. The data shows that districts C11, B2, B3, A7, E5, C6, E13, A1, B3, E13, and B3 have more than one victim.

	incident_numb...	district_...	district_name	multi_vict...
▶	222000342	C11	Dorchester	t
▶	222002059	B2	Roxbury	t
▶	222012303	B3	Mattapan	t
▶	222020180	A7	East Boston	t
▶	222024777	E5	West Roxbury	t
▶	222025588	C6	South Boston	t
▶	222025866	E13	Jamaica Plain	t
▶	222026988	A1	Downtown	t
▶	222035664	B3	Mattapan	t
▶	222036408	E13	Jamaica Plain	t
▶	222040222	B3	Mattapan	t